

2

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-222218

(P2000-222218A)

(43) 公開日 平成12年8月11日 (2000.8.11)

(51) Int.Cl.<sup>7</sup>

G 0 6 F 9/45

識別記号

F I

G 0 6 F 9/44

キーワード\* (参考)

3 2 2 F 5 B 0 8 1

審査請求 未請求 請求項の数 6 O L (全 10 頁)

(21) 出願番号

特願平11-23774

(22) 出願日

平成11年2月1日 (1999.2.1)

(71) 出願人

000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番1号

(72) 発明者

杉本 直史

東京都港区芝浦四丁目15番33号 株式会社富士通ビー・エス・シー内

(72) 発明者

青木 正樹

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

(74) 代理人

100089141

弁理士 岡田 守弘

最終頁に続く

(54) 【発明の名称】 コンパイル装置および記録媒体

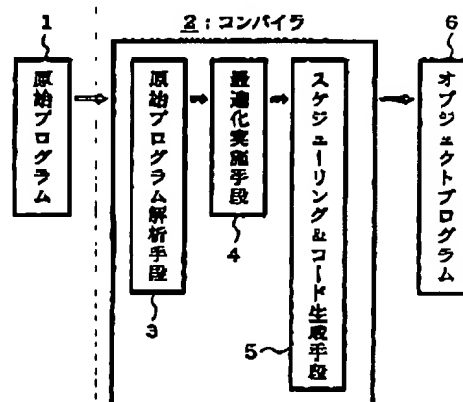
(57) 【要約】

(修正有)

【課題】 ソースプログラムの最適化を行うコンパイル装置および記録媒体に関し、コンパイルするときの高速化を図ると共に最適化を行い更に高速化を図る。

【解決手段】 ソースプログラム中から分岐を検出して削除する手段と、削除する分岐を実行した時の真あるいは偽を条件コードに設定させる命令を生成する手段と、削除する分岐を実行して分岐する場合の処理を実行させる第1の命令および分岐しない場合の処理を実行させる第2の命令を生成すると共に、条件コードの値に従い第1の命令あるいは第2の命令のいずれかを実行させ他の命令の実行を抑止させる第3の命令を生成し、第1の命令、第2の命令、第3の命令を少なくとも含む1つの同時実行可能な特定命令を生成する手段とを備える。

本発明のシステム構成図



【特許請求の範囲】

【請求項1】ソースプログラム中の分岐を削除して最適化を行うコンパイル装置において、  
ソースプログラム中から分岐を検出して削除する手段と、

上記削除する分岐を実行した時の真あるいは偽を条件コードに設定させる命令を生成する手段と、

上記削除する分岐を実行して分岐する場合の処理を実行させる第1の命令および分岐しない場合の処理を実行させる第2の命令を生成すると共に、上記条件コードの値に従い上記第1の命令あるいは上記第2の命令のいずれかを実行させ他の命令の実行を抑止させる第3の命令を生成し、上記第1の命令、上記第2の命令、上記第3の命令を少なくとも含む1つの同時実行可能な特定命令を生成する手段とを備えたことを特徴とするコンパイル装置。

【請求項2】if文の比較部分を実行したときの真あるいは偽を上記条件コードに設定、当該真の場合の処理を行う上記第1の命令、および当該偽の場合の処理を行う上記第2の命令としたことを特徴とする請求項1記載のコンパイル装置。

【請求項3】上記分岐のthen節ブロックおよびelse節ブロックを結合し全体ブロックをまとめて最適化したことを特徴とする請求項2記載のコンパイル装置。

【請求項4】上記ソースプログラム中の分岐について、試実行して分岐確率が所定値以下のときに上記分岐の削除を行うことを特徴とする請求項1から請求項3のいずれかに記載のコンパイル装置。

【請求項5】上記ソースプログラム中の分岐に削除指示が設定されていたときに、上記分岐の削除を行うことを特徴とする請求項1から請求項3のいずれかに記載のコンパイル装置。

【請求項6】ソースプログラム中から分岐を検出して削除する手段と、

上記削除する分岐を実行した時の真あるいは偽を条件コードに設定させる命令を生成する手段と、

上記削除する分岐を実行して分岐する場合の処理を実行させる第1の命令および分岐しない場合の処理を実行させる第2の命令を生成すると共に、上記条件コードの値に従い上記第1の命令あるいは上記第2の命令のいずれかを実行させ他の命令の実行を抑止させる第3の命令を生成し、上記第1の命令、上記第2の命令、上記第3の命令を少なくとも含む1つの同時実行可能な特定命令を生成する手段として機能させるプログラムを記録したコンピュータ読取可能な記録媒体、

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ソースプログラム中の分岐を削除して最適化を行うコンパイル装置および記録媒体に関するものである。

【0002】

【従来の技術】従来、if文には比較して真のときにthen節を実行した後に次のブロックにジャンプ(分岐)し、偽のときにelse節を実行するというように、例えば図6の(a)に示すようにプログラムが記述され、一般的に2個の分岐を含むものとして、コンパイルして実行可能なオブジェクトプログラムを生成していた。以下図6の構成について簡単に説明する。

【0003】図6は、従来技術の説明図を示す。図6の(a)は、if文のプログラム例を示す。ここで、左端はC言語プログラムを表し、中央の欄はアセンブラでのイメージを表し、右側はアセンブラでのイメージにおける操作内容を表す。

【0004】C言語のif文(if(x==1))で・真ならば、then節ブロックの処理を実行した後に、ジャンプ命令(goto文)で分岐(分岐①)して次のブロック(lab 003:)の処理に進む。

【0005】・偽ならば、else節ブロックに分岐(分岐)して処理を行い、続いて次のブロック(lab 003)の処理に進む。

以上の様子を模式的に表すと、図6の(b)に示すようになる。

【0006】図6の(b)は、(a)の模式図を示す。図6の(b)において、S11は、比較命令である。この比較命令は、図6の(a)のアセンブラでのイメージ中のcmp命令である。

【0007】・S12は、S11で真のときに、S12でthen節ブロックを実行し、分岐してS14の次のブロックの実行に進む。これは、上述した真ならばに相当する。

【0008】・S13は、S11で偽のときに、分岐してS13でelse節ブロックの処理を行い、S14の次のブロックの実行に進む。これは、上述した偽ならばに相当する。

【0009】

【発明が解決しようとする課題】上述した図6の(a)、(b)に示すように、if文を実行する場合には、分岐 と分岐 の2個のいずれか一方を必ず実行する必要があり、このままコンパイルした場合には当該2個の分岐を実行するためのオブジェクトコードを生成すると多くのステップ(例えば分岐以外の通常の命令が1つで実行するとすれば、約8つという多くのステップ)が必要となり、高速化を図れないという問題があった。

【0010】また、if文では上述したように2個の分岐があるため、当該if文が複数のブロックに分割されてしまい、ブロック内でしか最適化(例えばレジスタの最適化)しか図れず、もしif文の全部のブロックをまとめて最適化すれば、then節ブロックとelse節ブロックのいずれか一方を実行することで、更に最適化を進める(then節ブロックとelse節ブロックの

レジスタを共用して割当てなどの最適化を進める) ことができないという問題があった。

【0011】本発明は、これらの問題を解決するため、*if*文などにおける分岐を削除して複数命令を同時実行可能な特定命令(例えば*nilf*命令)中の条件コード(真あるいは偽)で指定された特定命令中の命令を実行し他の命令をマスクして実行を抑止し、*if*文などをコンパイルするときの従来の分岐を削除して高速化を図ると共に*then*節ブロックおよび*else*節ブロックなどを1つにまとめて最適化を行い更に高速化を図ることを目的としている。

【0012】

【課題を解決するための手段】図1を参照して課題を解決するための手段を説明する。図1において、原始プログラム(ソースプログラム)1は、C言語などで記述されたソースプログラムであって、ここでは、*if*文などの分岐を含むソースプログラムである。

【0013】コンパイラ2は、ソースプログラム1を最適化して実行可能形式のオブジェクトプログラム6を生成するものであって、ここでは、最適化実施手段4などから構成されるものである。

【0014】最適化実施手段4は、最適化を行うものであって、ここでは、分岐を削除して特定命令などに置き換えて最適化を行うものである。次に、動作を説明する。

【0015】最適化実施手段4がソースプログラム1中から分岐を検出して削除し、削除する分岐を実行した時の真あるいは偽を条件コードに設定させる命令を生成した後、削除する分岐を実行して分岐する場合の処理を実行させる第1の命令および分岐しない場合の処理を実行させる第2の命令を生成すると共に、条件コードの値に従い第1の命令あるいは第2の命令のいずれかを実行させ他の命令の実行を抑止させる第3の命令を生成し、第1の命令、第2の命令、第3の命令を少なくとも含む1つの同時実行可能な特定命令を生成し、コンパイルするようにしている。

【0016】この際、*if*文の比較部分を実行したときの真あるいは偽を条件コードに設定する命令を生成、真の場合の処理を行う第1の命令、および偽の場合の処理を行う第2の命令を生成するようにしている。

【0017】また、分岐の*then*節ブロックおよび*else*節ブロックを結合し全体ブロックをまとめて最適化するようにしている。また、ソースプログラム1中の分岐について、試実行して分岐確率が所定値以下のときに分岐の削除を行うようにしている。

【0018】また、ソースプログラム1中の分岐に削除指示が設定されていたときに、分岐の削除を行うようにしている。従って、*if*文などにおける分岐を削除して複数命令を同時実行可能な特定命令(例えば*nilf*命令)中の条件コード(真あるいは偽)で指定された当該

特定命令中の命令を実行し他の命令をマスクして実行を抑止することにより、*if*文などをコンパイルするときの従来の分岐を削除して高速化を図ると共に*then*節ブロックおよび*else*節ブロックなどを1つにまとめて最適化を行い更に高速化を図ることが可能となる。

【0019】

【実施例】次に、図1から図5を用いて本発明の実施の形態および動作を順次詳細に説明する。

【0020】図1は、本発明のシステム構成図を示す。図1において、原始プログラム(ソースプログラム)1は、C言語などで記述されたソースプログラムであって、例えば後述する図2の(a)の欄に示すように、C言語で記述したプログラムであり、ここでは、*if*文を含むプログラムである。

【0021】コンパイラ2は、ソースプログラム1を最適化して実行可能形式のオブジェクトプログラム6を生成するものであって、原始プログラム解析手段3、最適化実施手段4、およびスケジューリング&コード生成手段5などから構成されるものである。

【0022】原始プログラム解析手段3は、原始プログラム1を形態素解析および構文解析などを行い、*if*文などを解析してこれに続く処理の行い易い中間言語に変換するものである。

【0023】最適化実施手段4は、最適化を行うものであって、ここでは、分岐命令を削除して特定命令に置き換えて最適化を行ったり、所定ブロック単位にレジスタ割り当てなどの最適化を行ったりなどするものである。

【0024】スケジューリング&コード生成手段5は、最適化を行った後に、公知のスケジューリングを行い、更に実行可能形式のコード(機械語レベルのコード)を生成するものである。この生成されたものがオブジェクトプログラム6となる。

【0025】オブジェクトプログラム6は、実行可能形式のプログラムである。図2は、本発明の具体例(図6の(a)の分岐命令削除)を示す。これは、既述した図6の(a)のC言語プログラムのアセンブラでのイメージ中に記述した分岐①および分岐②を削除し、その代わりに図2の(a)中に記述し、図2の(b)に示すように動作させると、図6の(a)中の分岐①および分岐②を削除しても同じ処理を実行することが可能となる。

【0026】詳述すれば、図2の(a)のアセンブラでのイメージ中に記述した、

```
・条件コードに設定: cmp x, 1, ccr0
・特定命令
・第1のスロット: add gr0, 10, gr18
・第2のスロット: add gr0, 20, gr18
・第3のスロット: nop
・第4のスロット: nilf ccr0, 1, ccr0, 0, true, 1
・命令: store gr18, a
```

によって、図6の(a)の従来の分岐①および分岐②を削除しても同じ処理を実行させることが可能となる。

【0027】ここで、

・条件コードの設定(`cmp x, 1, ccr0`)によって、図2の(a)の操作内容欄に記述したように、`x`と1を比較し、条件コード`ccr0`に真のときに1を設定し、偽のときに0を設定する。

【0028】・特定命令(1行で第1から第4のスロットの命令を同時実行できる命令)によって、図2の(b)に記載したように、第4のスロットの条件コード`ccr0`が1の場合には、第1のスロットを実行し、ここでは、`gr0`に10を加算して`gr18`に格納する。

【0029】・第4のスロットの条件コード`ccr0`が0の場合には、第2のスロットを実行し、ここでは、`gr0`に20を加算して`gr18`に格納する。第4のスロットの条件コード`true`が1(常に行う)の場合には、第3のスロットを実行し、ここでは、`nop`を実行する。

【0030】・最後の命令(`store gr18, a`)によって、`gr18`の内容を`a`に格納し、一連の処理を終了し、次の命令の処理に進む。以上の特定命令(例えば`nlif`命令)の第4のスロットの条件コードによって第1のスロットの命令あるいは第2のスロットの命令のいずれか一方のみが実行され、他方がマスクされて実行が抑止されるので、分岐が真あるいは偽に対応して第1のスロットの命令あるいは第2のスロットの命令のいずれか一方のみが自動的に実行されることとなる。これにより、従来の図6の(a)の分岐①および分岐②を削除しても、上記した、条件コードの設定、特定命令、命令を設定することで、同じ処理を分岐なしに実行することが可能となる。

【0031】尚、図2の(a)のアセンブラでのイメージ欄中の特定命令の`&`は1命令の継続を表し、ここでは、4行で記述した4命令で1つの特定命令となり、例えば1行で同時実行できる命令である。

【0032】図3は、本発明の説明図(図2の(a)の模式図)を示す。これは、既述した図2の(a)のアセンブラでのイメージを模式的に表したものである。図3において、`S1`は、条件コード設定を行う。これは、図2の(a)のアセンブラでのイメージ欄の・`cmp x, 1, ccr0`(`x`と1を比較し、真あるいは偽を`ccr0`に設定する)に対応する。

【0033】`S2`は、`then`節ブロック、`else`節結合ブロックとする。これは、図2の(a)の`then`節ブロックと`else`節ブロックとを結合して1つのブロックとし、最適化処理を行う。例えば1つのブロックに対して、変数にレジスタ割付を行う。この際、従来は、`then`節ブロック、`else`節ブロック毎に個別に変数にレジスタ割付を行っており、`then`節ブロッ

クと`else`節ブロックはいずれか一方のみしか実行されないで、いずれか一方のレジスタ割付したレジスタが無用に割り付けていたこととなる。しかし、本願発明では、`then`節ブロックと`else`節ブロックとを1つのブロックにしてレジスタ割付を行うので、割り付けるレジスタ数を削減して他にそのレジスタを割り付けより高速実行可能な最適化を行うことが可能となる。

【0034】`S3`は、次のブロックの処理を行う。以上のように、従来は既述した図6の(b)に示すように、分岐①と分岐②があり分岐による速度低下があると共に、`then`節ブロックと`else`節ブロックとを独立に最適化を行う必要があり、レジスタ割付では多くのレジスタを割り付けざるを得なかったが、本願発明では、図3に示すように、分岐①と分岐②とを削除して高速実行を図ることが可能となり、かつ`then`節ブロックと`else`節ブロックとを結合した1つのブロックに対して最適化を行い、より高速実行を図ることが可能となる。

【0035】図4は、本発明の説明図(分岐命令削除指定)を示す。これは、C言語プログラム中の分岐を持つ例えば`if`文の直前行に図示(a)に示すように、分岐命令を削除する指定を行なった様子を示す。この場合には、当該分岐命令削除指定に続く分岐を含む、ここでは、`if`文の分岐を自動的に削除し、既述した図2の(a)のアセンブラでのイメージ欄に示すようにする。

【0036】以上のように、ソースプログラム1中で分岐命令削除指定を記述することで、自動的に`if`文などに含まれる分岐を削除し、条件コードの設定、特定命令、命令などを挿入し、実行速度の高速化および最適化ブロックを広くしてより最適化を進めて高速実行可能なオブジェクトプログラム6を自動生成することが可能となる。

【0037】図5は、本発明の入れ子の`if`の展開例を示す。これは、既述した図2の(a)の`if`文が`then`節ブロックと`else`節ブロックとからなっていたが、更に、`then`節ブロック、`else`節ブロック内でそれぞれ`if`文が記述され、いわゆる入れ子となったときの様子を示したものである。この入れ子になった場合には、入れ子のなる前の条件コードを当該入れ子の内部では継承する必要があるので、図示のようにすることで、入れ子の`if`文についても同様に、既述した従来の図6の(a)、(b)の分岐①と分岐②を削除することが可能となる。

【0038】図5において、図中のアセンブラでのイメージ欄の各命令は左側に記載した

・外`then` :  
 ・外`then`—内`then` :  
 ・外`then`—内`else` :  
 ・外`else` :  
 ・外`else`—内`then` :

・外 `else` - 内 `else` :

という順番に入れ子となっている。

【0039】また、図中の①から ④は、特定命令を表し、既述した図2の (b) に示すと同様である。例えばの特定命令は、

- ・第1のロット: `cmp y, 2, ccr2`
- ・第2のロット: `nop`
- ・第3のロット: `nop`
- ・第4のロット: `nif ccr0, 1, true, 1, true, 1`

であって、第4のロットの命令をもとに (図2の (b) 参照)、操作内容の欄に記載したように、`ccr0` が1の場合に `y` と `2` を比較し、`ccr2` に条件コードを設定することを表す。以下 から の特定命令について同様に、操作内容の欄に記載したように処理を行うことで、`if` 文の `then` 節ブロック、`else` 節ブロックにそれぞれ `if` 文があってもそれぞれの分岐を削除し、図5の全体を1つのブロックに結合し、分岐を削除したことで高速実行を図ることが可能となると共に、全体を1つのブロックとして最適化を行い、より高速化を図ることが可能となる。

【0040】

【発明の効果】以上説明したように、本発明によれば、`if` 文などにおける分岐を削除して複数命令を同時実行

可能な特定命令 (例えば `nif` 命令) 中の条件コード

(真あるいは偽) で指定された当該特定命令中の命令を実行し他の命令をマスクして実行を抑止する構成を採用しているため、`if` 文などをコンパイルするときの従来の分岐を削除して高速化を図ると共に `then` 節ブロックおよび `else` 節ブロックなどを1つにまとめて最適化を行い更に高速化を図ることが可能となる。

【図面の簡単な説明】

【図1】本発明のシステム構成図である。

【図2】本発明の具体例 (図6の (a) の分岐命令削除) である。

【図3】本発明の説明図 (図2の (a) の模式図) である。

【図4】本発明の説明図 (分岐命令削除指定) である。

【図5】本発明の入れ子の `if` の展開例である。

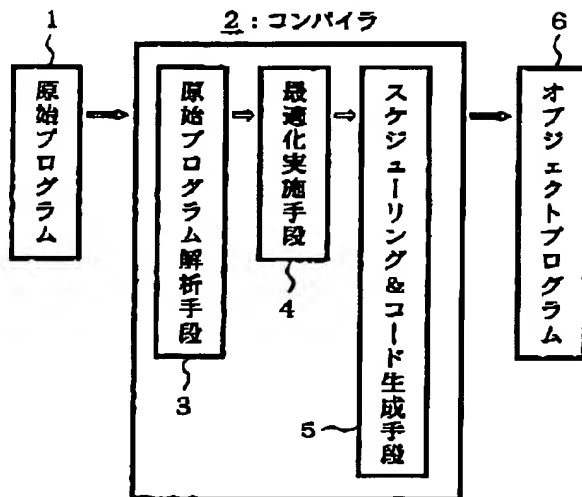
【図6】従来技術の説明図である。

【符号の説明】

- 1: 原始プログラム (ソースプログラム)
- 2: コンパイラ
- 3: 原始プログラム解析手段
- 4: 最適化実施手段
- 5: スケジューリング&コード生成手段
- 6: オブジェクトプログラム

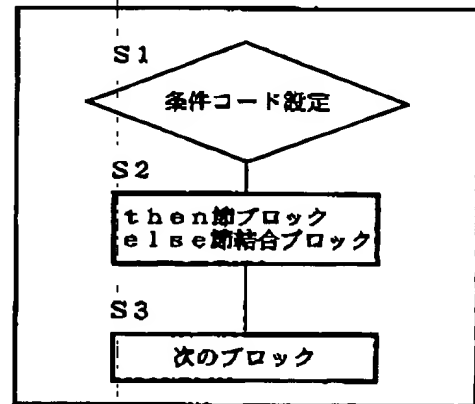
【図1】

本発明のシステム構成図



【図3】

本発明の説明図 (図2の (a) の模式図)



(6)

特開2000-222218

【図2】

本発明の具体例 (図6の(a)の分岐命令削除)

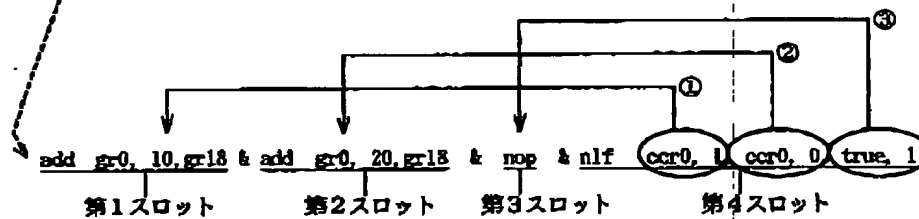
(a)

| C言語プログラム  | アセンブラでのイメージ   | 操作内容  |
|---|---|---|
| <pre> if(x == 1) {     a=10; } else{     a=20; } </pre> | <pre> lab_000:     cmp    x, 1, ccr0      add    gr0, 10, gr18 &amp;     add    gr0, 20, gr18 &amp;     nop    &amp;     nlf    ccr0, 1, ccr0, 0, true, 1      store  gr18, a lab_001: </pre> | <p>xと1を比較。<br/>ccr0に条件コードを設定する。</p> <p>ccr0が1なら10をgr18に入れる。<br/>ccr0が0なら20をgr18に入れる。</p> <p>各スロットを制御する命令。<br/>※ &amp; は1命令の継続を表す。</p> <p>gr18の内容をaに入れる。</p> |

(・アンダーライン: 分岐命令を表す。  
・lab\_00X: 処理ブロックを表す。)

←1命令(nlf命令)

(b) nlf命令の説明



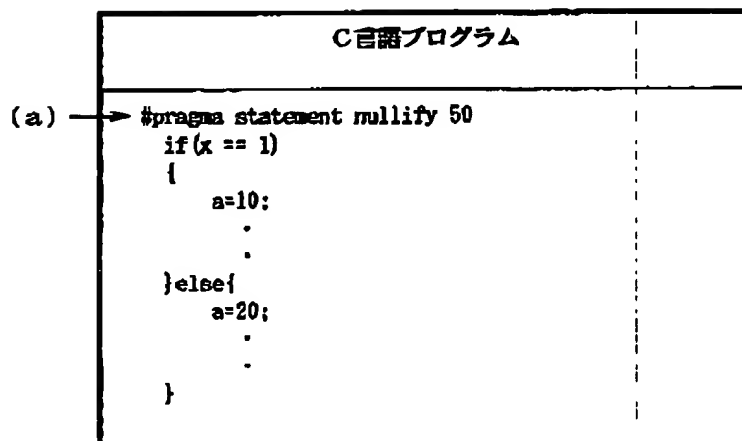
nlf命令(第4スロット)は、各スロット(第1～3スロット)の命令の制御を行う。  
 ①第1スロットを制御する。ccr0が1の場合、{add gr0, 10, gr18}を実行する。  
 ②第2スロットを制御する。ccr0が0の場合、{add gr0, 20, gr18}を実行する。  
 ③第3スロットを制御する。nop命令を常に実行する。  
 (逆の意味として、true, 0(常に実行しない。)もある。)

(7)

特開2000-222218

【図4】

本発明の説明図（分岐命令削除指定）



(8)

特開2000-222218

【図5】

本発明の入れ子のifの展開例

|               | C言語プログラム     | アセンブラでのイメージ   | 操作内容  |
|---------------|--------------|---|---|
| 外-then        | if(x == 1) { | lab_000:<br>cmp x, 1, ccr0  | xと1を比較。<br>ccr0に条件コードを設定する。   |
| 外-then-内-then | if(y == 2) { | xor ccr0, -1, ccr1<br>① cmp y, 2, ccr2 &<br>nop & nop &<br>nlif ccr0, 1, true, 1, true, 1     | ccr0の排他的論理和をccr1に入れる。<br>ccr0が1の場合yと2を比較し、ccr2に条件コードを設定する。                              |
| 外-then-内-else | else {       | ② xor ccr2, -1, ccr3 &<br>nop & nop &<br>nlif ccr0, 1, true, 1, true, 1                       | ccr0が1の場合ccr2の排他的論理和をccr3に入れる。  |
| 外-else        | else {       | ③ and ccr0, ccr2, ccr2 &<br>nop & nop &<br>nlif ccr0, 1, ccr0, true, 1                        | ccr0が1の場合、外側のifから受け継いだccr0と内側のif(y=2)の条件コードのアンドをそれぞれ行う。                                 |
| 外-else-内-then | if(y == 3) { | ④ add gr0, 10, gr18 &<br>add gr0, 20, gr18 &<br>nop & nop &<br>nlif ccr2, 1, ccr3, 1, true, 1 | ccr2が1なら10をgr18に入れる。<br>ccr3が1なら20をgr18に入れる。<br>nlifは各スロットを制御する命令。<br>※ & は 1 命令の重複を表す。 |
| 外-else-内-else | else {       | ⑤ cmp y, 3, ccr4 &<br>nop & nop &<br>nlif ccr1, 1, true, 1, true, 1                           | ccr1が1の場合yと3を比較し、ccr4に条件コードを設定する。   |
|               | a=30;        | ⑥ xor ccr4, -1, ccr5 &<br>nop & nop &<br>nlif ccr1, 1, true, 1, true, 1                       | ccr1が1の場合ccr4の排他的論理和をccr5に入れる。  |
|               | a=40;        | ⑦ and ccr1, ccr4, ccr4 &<br>nop & nop &<br>nlif ccr1, 1, ccr1, 1, true, 1                     | ccr1が1の場合、外側のifから受け継いだccr1と内側のif(y=3)の条件コードのアンドをそれぞれ行う。                                 |
|               | }            | ⑧ add gr0, 30, gr18 &<br>add gr0, 40, gr18 &<br>nop & nop &<br>nlif ccr4, 1, ccr5, 1, true, 1 | ccr4が1なら30をgr18に入れる。<br>ccr5が1なら40をgr18に入れる。  |
|               | }            | store gr18, a   | gr18の内容をaに入れる。  |
|               | }            | lab_001:  |   |

・ lab\_001: 処理ブロックを表す。

・ ①～⑧: n & f 命令

(9)

特開 2000-222218

【図6】

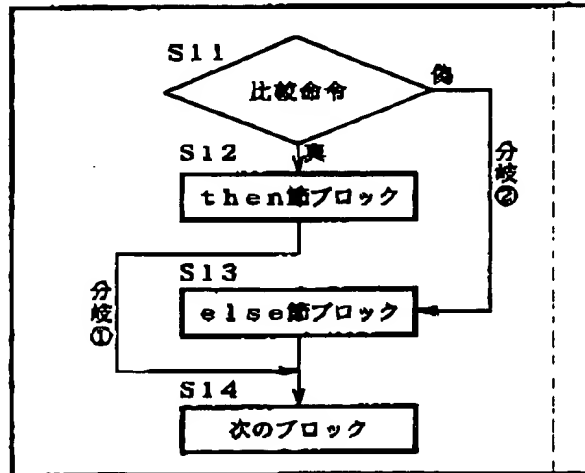
従来技術の説明図

(a)

| C言語プログラム   | アセンブラでのイメージ   | 操作内容  |
|--|---|---|
| <pre> if (x == 1) {     a=10; } else{     a=20; }                     </pre> | <pre> lab_000:     cmp    x,1, ccr0     br     ccr0, lab_002  lab_001:     store a, 10     goto  lab_003  lab_002:     store a, 20  lab_003:                     </pre> | <p>xと1を比較。<br/>ccr0に条件コードを設定する。<br/>比較結果が真ならlab_001に制御が移り、偽ならlab_002に制御が移る。</p> <p>分岐②<br/>Aに10を入れる。<br/>lab_003に制御を移す。</p> <p>分岐①<br/>Aに20を入れる。<br/>lab_003に制御を移す。</p> |

- ・ ccrX: 比較命令で設定された条件コードを保持する専用レジスタ (1bit)
- ・ アンダーライン: 分岐命令を表す
- ・ lab\_00X: 処理ブロックを表現する

(b) (a) の様式図



フロントページの続き

(72) 発明者 鈴木 清文  
神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(72) 発明者 高原 浩二  
神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(72) 発明者 山中 豊  
神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(72) 発明者 小泉 治道  
静岡県静岡市南町18番1号 株式会社富士  
通静岡エンジニアリング内

(10)

特開2000-222218

(72) 発明者 森島 政人  
静岡県静岡市南町18番1号 株式会社富士  
通静岡エンジニアリング内

(72) 発明者 山本 賢一  
静岡県静岡市南町18番1号 株式会社富士  
通静岡エンジニアリング内

(72) 発明者 谷村 森伸  
静岡県静岡市南町18番1号 株式会社富士  
通静岡エンジニアリング内

(72) 発明者 日下部 明  
静岡県静岡市南町18番1号 株式会社富士  
通静岡エンジニアリング内

(72) 発明者 山地 延佳  
神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

Fターム(参考) 5B081 AA06 GC22 GC23

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**